**CS 355: Topics in Cryptography** **Spring 2019**

# Lecture 5: Proofs of Knowledge, Schnorr's protocol, NIZK

Dima Kogan

## Recap

Last lecture we saw that languages in NP have zero knowledge proofs (if commitments exist). We'll start by a slightly differet view of NP. We say that a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ is an NP relation if:

- It is polynomially bounded: there exists a polynomial $p$, such that $|x| \leq p(|y|)$ for every $(x,y) \in \mathcal{R}$.
- There exists a deterministic polynomial time algorithm $V$ such that $(x,y) \in \mathcal{R} \iff V(x,y) = 1$.

The corresponding language $\mathcal{L}_R = \{y : (x,y) \in \mathcal{R}\}$ is called an NP language. A zero knowledge proof system for an NP relation $\mathcal{R}$ is a pair of interactive and randomized algorithms $P$, $V$, with the following properties:

- Completeness: $y \in \mathcal{L}_\mathcal{R} \Rightarrow \Pr\left[\langle P, V\rangle(y) = 1\right] \geq 2/3$.
- Soundness: $y \notin \mathcal{L}_\mathcal{R} \Rightarrow \forall P^* : \Pr\left[\langle P^*, V\rangle(y) = 1\right] \leq 1/3$.
- Zero knowledge: $\forall V^* \exists$ efficient $S$ such that $\forall y \in \mathcal{L} : \left\{\mathsf{Sim}(y)\right\} \approx_c \left\{\mathsf{View}_{V^*}(\langle P, V^*\rangle(y)\right\}$.

The third property guarantees that the verifier learns nothing from the interaction, except that $y \in \mathcal{L}$, since there exists an efficient simulator that can generate transcripts that are indistinguishable from the real ones.

## 1 Proofs of Knowledge

Note that the soundness requirement assures the verifier that if it accepts some statement $y$, then with high probability $y \in \mathcal{L}_\mathcal{R}$. Sometimes this assurance is not strong enough. For example, a login protocol in which the prover merely convinces the verifier that *there exists* a secret key corresponding to the verification key held by the verifier would not be intuitively secure. We would like the prover to convince the verifier that it actually *knows* the secret key.

**Definition 1.1.** An interactive proof system $P$,$V$ for an NP relation $\mathcal{R}$ is a **proof of knowledge with knowledge error** $\epsilon$, if there exists an algorithm $\mathcal{E}$, called an extractor, that runs in expected polynomial time[1], and such that for every $y$ and every prover $P^*$:

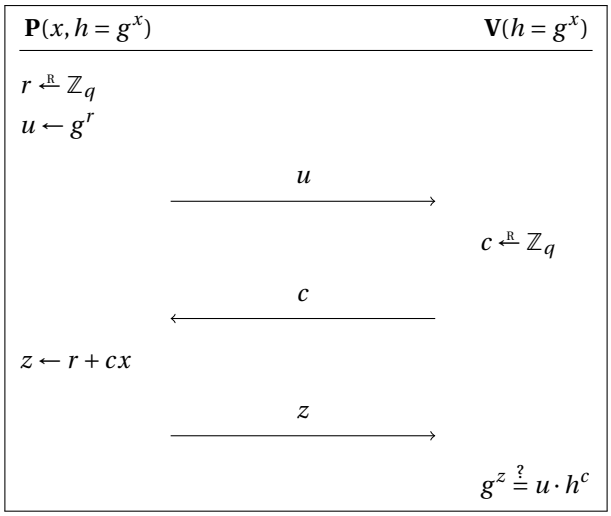
$$\Pr\left[(x,y) \in \mathcal{R} : x \leftarrow \mathcal{E}^{P^*}(y)\right] \geq \Pr\left[\langle P^*, V\rangle(y) = 1\right] - \epsilon\,.$$

The notation $\mathcal{E}^{P^*}$ means that $\mathcal{E}$ is an algorithm that gets black-box access to the algorithm $P^*$, including the power to rewind the prover. The probability $\epsilon$ is called the knowledge error of the proof system. Note that if a proof system has knowledge error $\epsilon$, it means that it has soundness error of *at most* $\epsilon$.

[1] An algorithm is said to run in *expected polynomial time* if there exists a polynomial $p$ such that for every $x$, the expected running time of the algorithm on $x$ is at most $p(|x|)$, where the expectation is taken over the random choices made by the algorithm.

## 2 Schnorr's Protocol: Proof of Knowledge of Discrete Log

Suppose that a prover wants to prove it knows the discrete logarithm $x$ of some group element $h = g^x \in \mathbb{G}$, where $\mathbb{G}$ is a group of prime order $q$. Here $\mathcal{R} = \{(x,h) \in \mathbb{Z}_q \times \mathbb{G} : g^x = h\}$, where the group $\mathbb{G}$ and the generator $g$ are public parameters.

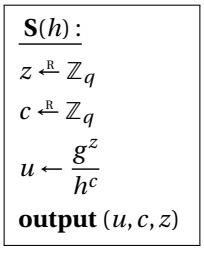



**Completeness:** if $z = r + cx$, then $g^z = g^{r+cx} = g^r \cdot \left(g^x\right)^c = u \cdot h^c$.

**Honest-verifier zero knowledge (HVZK):** for every $g, h \in \mathbb{G}$, the output of the simulator needs to be indistinguishable from the distribution of the transcripts

$$\{\mathsf{View}_V(P(x,h) \leftrightarrow V(h))\} = \left\{(g^r, c, r+cx) : r, c \xleftarrow{R} \mathbb{Z}_q\right\} = \left\{(u,c,z) : c, z \xleftarrow{R} \mathbb{Z}_q,\ g^z = u \cdot h^c\right\}$$

We construct a simulator that output the same distribution by running the protocol "in reverse":




Since $z$ is chosen at random, then the resulting $u$ is random, and the output is *distributed identically* as the real transcript.

Why does this proof not give zero knowledge, but only HVZK? If some dishonest verifier $V^*$ chooses $c$ adaptively (and not uniformly at random), then the above simulation is no longer indistinguishable from the real transcript.

**Proof of knowledge:** Let $P^*$ be a (possibly malicious) prover that convinces the honest verifier with probability $\delta$. For simplicity, we give here the proof only for the case $\delta = 1$. (For the general case, see Boneh-Shoup Chapter 19.1.) We construct the extractor as follows:

$\mathcal{E}^{P^*}(h)$ :

1 : Run the prover $P^*$ to obtain an initial message $u$.
2 : Send a random challenge $c_1 \xleftarrow{R} \mathbb{Z}_q$ to $P^*$ and get a response $z_1$.
3 : Rewind the prover $P^*$ to its state after the first message.
4 : Send it another random challenge $c_2 \xleftarrow{R} \mathbb{Z}_q$ and get a response $z_2$.
5 : Compute and output $x = \dfrac{z_1 - z_2}{c_1 - c_2} \in \mathbb{Z}_q$.

Since $P^*$ succeeds with probability 1, we know that

$$g^{z_1} = u \cdot h^{c_1} \quad \text{and} \quad g^{z_2} = u \cdot h^{c_2}.$$

Therefore

$$\frac{g^{z_1}}{h^{c_1}} = \frac{g^{z_2}}{h^{c_2}} \quad \Rightarrow \quad g^{z_1 - z_2} = h^{c_1 - c_2} \quad \Rightarrow \quad h = g^{\frac{z_1 - z_2}{c_1 - c_2}} \quad \Rightarrow \quad x = \frac{z_1 - z_2}{c_1 - c_2}.$$

Note that the extraction fails if $c_1 = c_2$, which happens with probability $1/q$. Therefore, the knowledge error here is $\epsilon = 1/q$.

## 3 Non-Interactive Zero Knowledge

Can we construct a zero-knowledge proof system in which the proof is a *single* message from the prover to the verifier? The zero-knowledge definition immediately implies that a simulator can simulate a message indistinguishable from the real proof for every $y \in \mathcal{L}$, whereas by soundness, for every $y \notin \mathcal{L}$ the proof $\mathsf{Sim}(y)$ must be rejected by the verifier. Therefore $V(\mathsf{Sim}(y)) = 1 \iff y \in \mathcal{L}$, and we get an efficient algorithm to decide $\mathcal{L}$. (In complexity-theoretic terms, we say that $\mathcal{L} \in \mathsf{BPP}$.) The conclusion is that one-round proofs only exist in a trivial case.

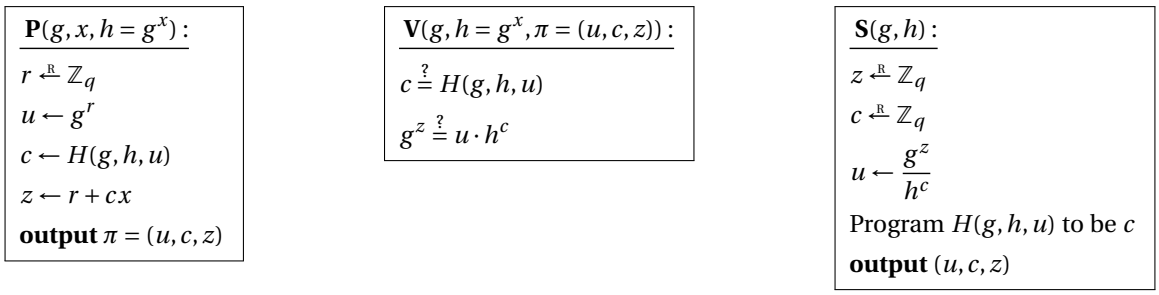
Intuitively, it looks like a dead end, since the simulator and the real prover stand on equal grounds, so we cannot get something non-trivial from the prover. To circumvent this, we move to the random oracle model, in which:

- The prover and the verifier both have access to a hash function modeled as a random oracle.
- The simulator can program the random oracle. The adversary, that tries to distinguish between the real transcript and the simulated one, can make oracle queries, but the simulator chooses the responses as well.

Note that although the simulator can program the random oracle any way it wants, the values it chooses for the responses must be indistinguishable from random, since otherwise the adversary could distinguish just by making oracle queries.

### 3.1 The Fiat-Shamir Heuristic

The Fiat-Shamir heuristic is a technique to convert an interactive protocol to a non-interactive proof in the random oracle model. The key idea is that we replace the verifier's random challenge with the value of a hash function (which we model as a random oracle) on the prover's first message and the input. For example, for the Schnorr identification protocol, we obtain the following:




**Security:**

- Completeness: same as in the interactive Schnorr protocol.
- Zero knowledge: the simulator proceeds as in the interactive protocol, only that now, the challenge is not a message from the verifier, but rather a value of the hash function. The simulator programs the random oracle to have the value $c$ it has chosen at the point $(g, h, u)$ (see figure above).
- Knowledge: Similar to the original extractor, only now, when rewinding the prover, the extractor changes the value of the random oracle to obtain two different transcripts with the same commitment, from which it can extract the discrete log $x$ of $h$.

### 3.2 Signatures

If we bind the NIZK proof to a specific message by adding the message as an additional input to the hash function (random oracle)

$$c \xleftarrow{R} H(g, h, u, m),$$

we obtain a signature scheme, in which $sk = (x)$ and $pk = (h = g^x)$. We can then prove the security of the resulting signature scheme (existenital unforgeability) in the random oracle model, by showing that we can use an adversary that forges a signature to construct another adversary that breaks Schnorr's interactive protocol.